

Finding and Fixing Your Errors

Charles Severance

Textbook: Build Your own Ruby on Rails Application by Patrick Lenz (ISBN:978-0-975-8419-5-2)

Common Errors and Mistakes

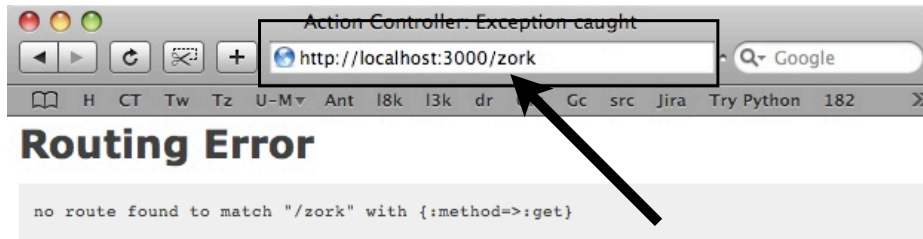
When bad syntax happens to good people.

Outline

- Using and Configuring jEdit
- URL Related mistakes - “No Route to...”
- View Errors - Mistakes in Embedded Ruby
- Controller Errors - Mistakes in Ruby
- The dreaded “expecting kEND” error
- Tips and Tricks

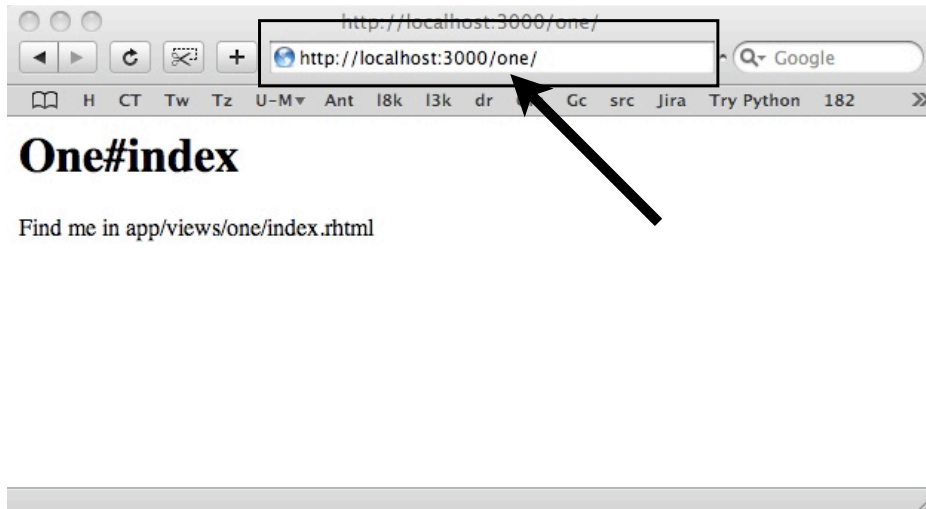
jEdit - www.jedit.org

- These examples assume the use of jEdit and the RubyPlugin
 - www.jedit.org



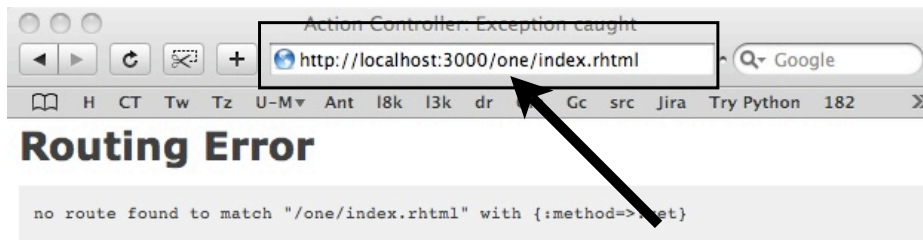
The first parameter after the :3000/ is the name of the controller. In this case I mistakenly used “zork” - the application name.

“No route found” is saying - “Looking at this URL, I see no controller named zork - so I have no clue where to go”.



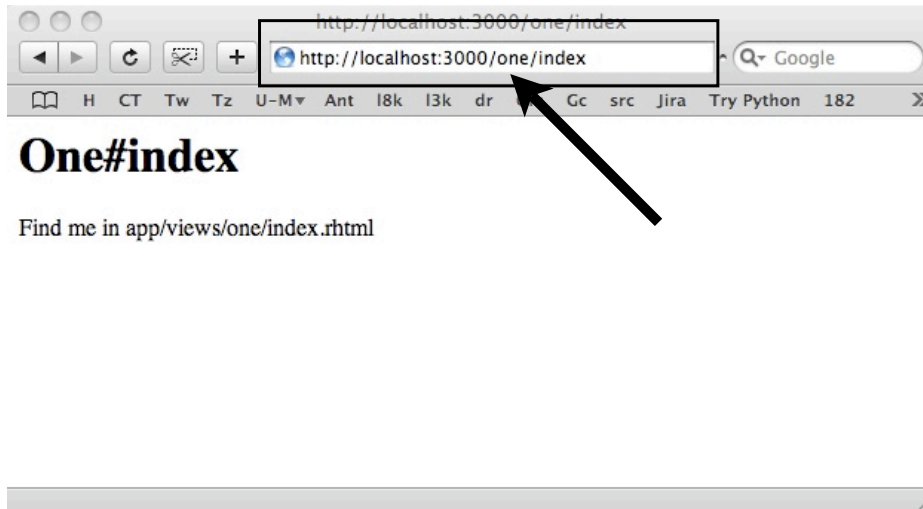
The fix is to use the name of your controller as the first thing after the host and port.

`http://localhost:3000/one/`



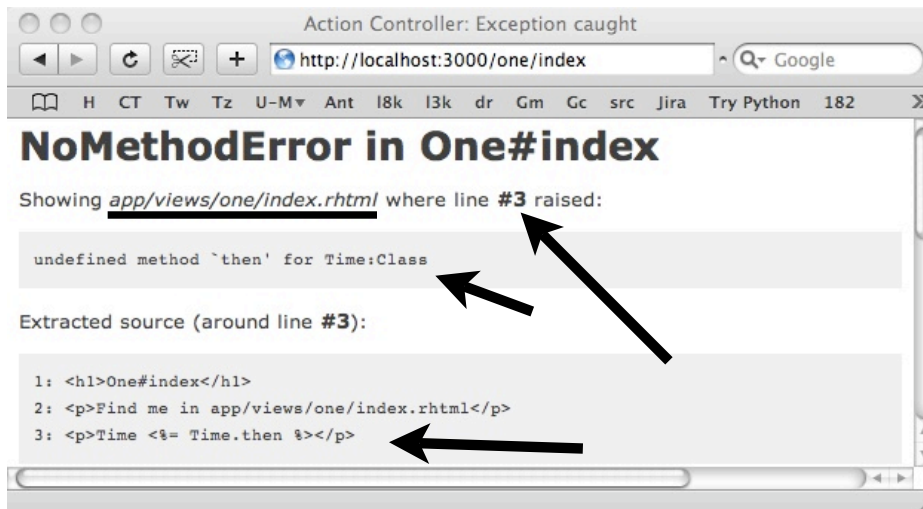
After the controller (one) the next parameter is an action. `index.rhtml` is the view associated with the “index” action.

The rails system takes URLs and hands them to actions - it is not serving files. Files in Rails applications come from the public folder and have a different path.

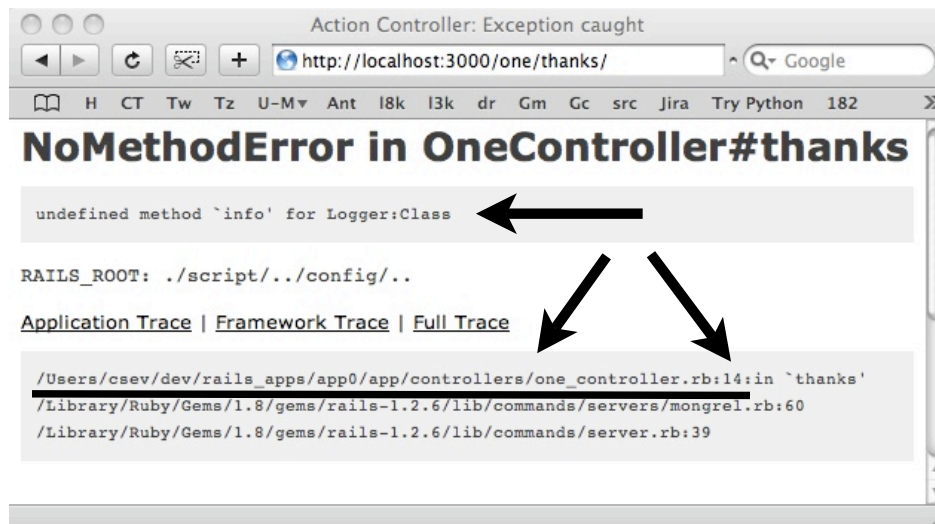


Now you are sending the request to the action named “index” in the controller named “one”.

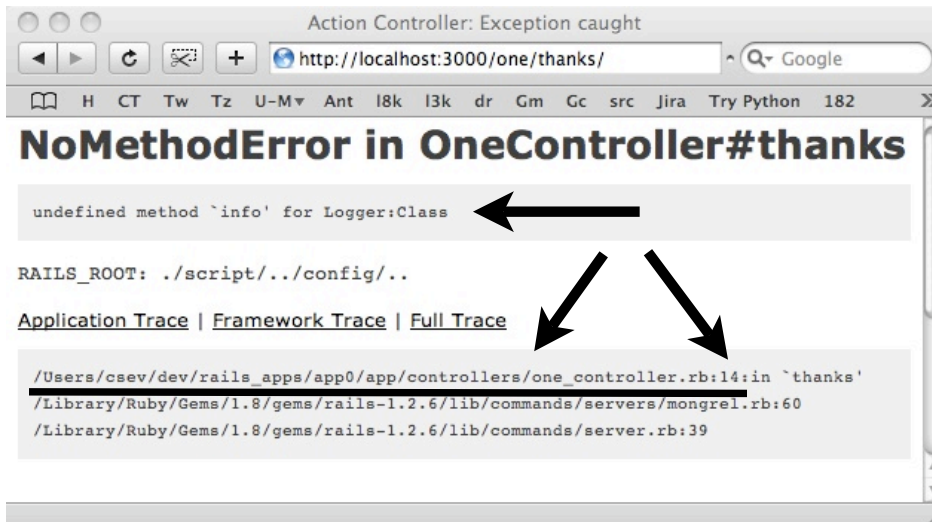
The text is coming from the file “index.rhtml” but that is happening after the controller processes the action and then triggers the view.



This is telling you that you have an error in a file (I) index.rhtml - it even tells you the path of the file within the application. It tells you what is wrong (as it imagines it) and what line it detected the error. This one is easy. It should be Time.now instead of Time.then.

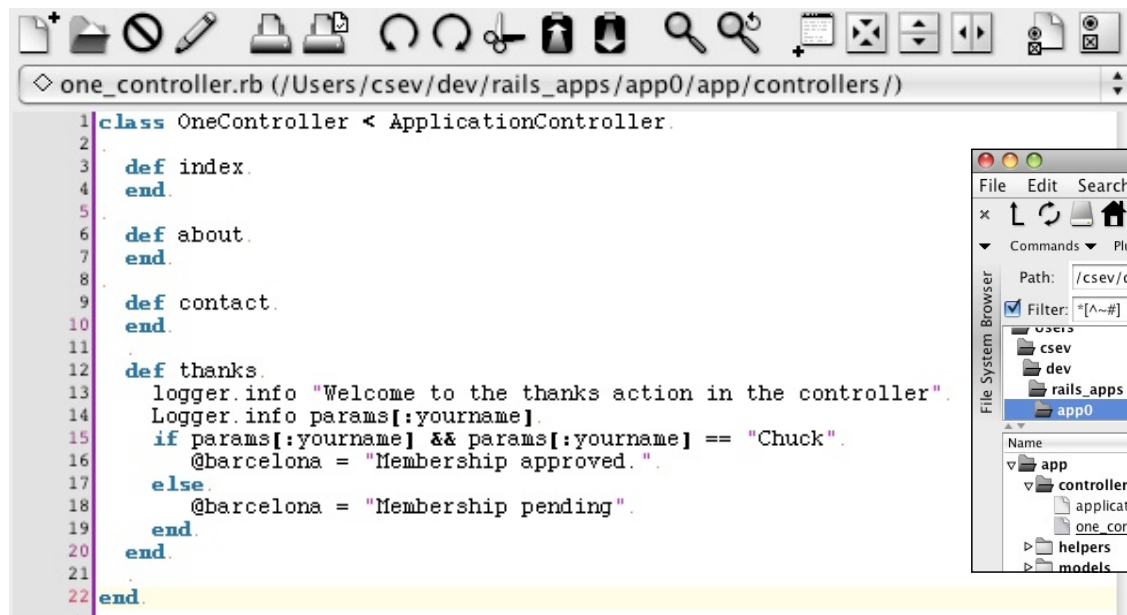


This is an error in the controller - it tells you the file name (underlined) and the line number in the file. It also tells you (undefined method) what it thinks is wrong.

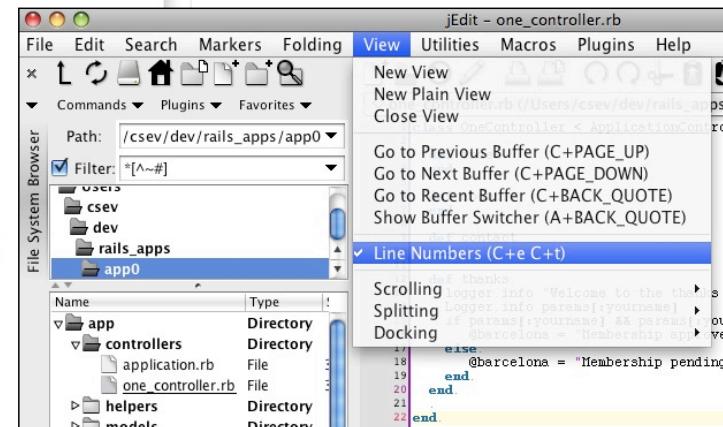


Capitalized by
mistake on line 14.

```
def thanks  
  logger.info "Welcome to the thanks action in the controller"  
  Logger.info params[:yourname]  
  if params[:yourname] && params[:yourname] == "Chuck"  
    @barcelona = "Membership approved."  
  else  
    @barcelona = "Membership pending"  
  end  
end
```

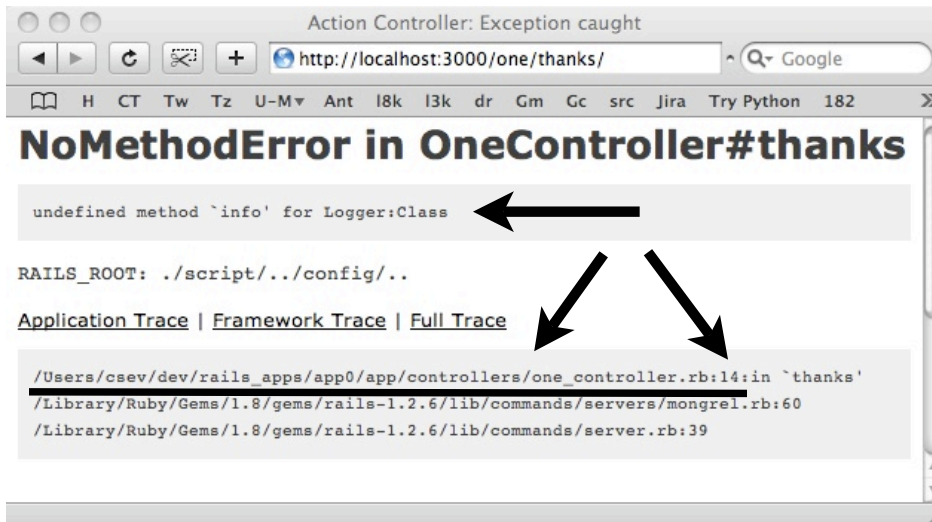


```
1 class OneController < ApplicationController.  
2  
3   def index.  
4     end.  
5  
6   def about.  
7     end.  
8  
9   def contact.  
10    end.  
11  
12  def thanks.  
13    logger.info "Welcome to the thanks action in the controller".  
14    Logger.info params[:yourname].  
15    if params[:yourname] && params[:yourname] == "Chuck".  
16      @barcelona = "Membership approved.".  
17    else.  
18      @barcelona = "Membership pending".  
19    end.  
20  end.  
21  
22 end.
```



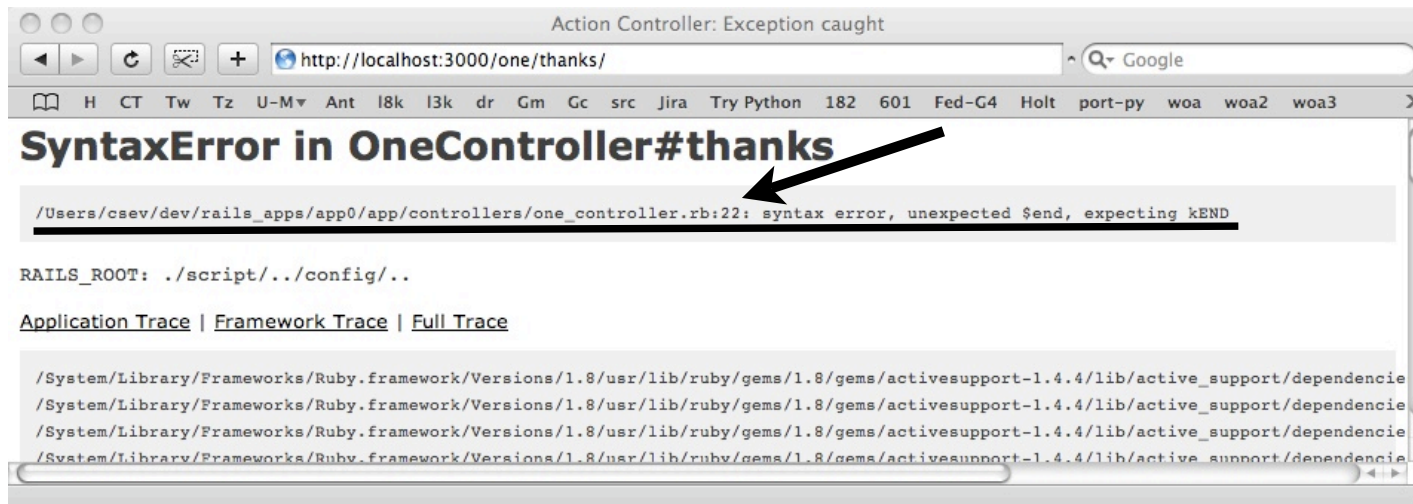
jEdit Tip: Show line numbers while editing.

View -> Line Numbers



Capitalized by
mistake on line 14.

```
def thanks
  logger.info "Welcome to the thanks action in the controller"
  Logger.info params[:yourname]
  if params[:yourname] && params[:yourname] == "Chuck"
    @barcelona = "Membership approved."
  else
    @barcelona = "Membership pending"
  end
end
```



This “expecting kEND” error is the single largest time waster of of students. Often it results in hours of flailing around. Many times the only solution is to wait several days until lab at which point the instructor solves it in 30 seconds. This means that you very much appreciate the nice instructor - but you decide to hate computers and decide that you will just hire programmers from this point on!

Action Controller: Exception caught

http://localhost:3000/one/thanks/

SyntaxError in OneController#thanks

`/Users/csev/dev/rails_apps/app0/app/controllers/one_controller.rb:22: syntax error, unexpected $end, expecting kEND`

RAILS_ROOT: ./script/../config/..

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/lib/ruby/gems/1.8/gems/activesupport-1.4.4/lib/active_support/dependencie
/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/lib/ruby/gems/1.8/gems/activesupport-1.4.4/lib/active_support/dependencie
/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/lib/ruby/gems/1.8/gems/activesupport-1.4.4/lib/active_support/dependencie
/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/lib/ruby/gems/1.8/gems/activesupport-1.4.4/lib/active_support/dependencie

The most important thing to remember is that the line number in these messages is when Rails detects the error. It is not necessarily the line on which you **made** the error.

```

1 class OneController < ApplicationController.
2
3   def index.
4
5
6   def about.
7     end.
8
9   def contact.
10    end.
11
12   def thanks.
13 logger.info "Welcome to the thanks action in the controller".
14 logger.info params[:yourname]
15     if params[:yourname] && params[:yourname] == "Chuck".
16 @barcelona = "Membership approved.".
17     else.
18     @barcelona = "Membership pending".
19     end.
20   end.
21
22 end.

```

Which would you rather read?

We are not forced to neatly indent our code - Ruby is pretty tolerant of whitespace. But for us to read our own code - indenting makes a lot of sense.

```

1 class OneController < ApplicationController.
2
3   def index.
4
5
6   def about.
7     end.
8
9   def contact.
10    end.
11
12   def thanks.
13     logger.info "Welcome to the thanks action in the controller".
14     logger.info params[:yourname].
15     if params[:yourname] && params[:yourname] == "Chuck".
16       @barcelona = "Membership approved.".
17     else.
18       @barcelona = "Membership pending".
19     end.
20   end.
21
22 end.

```

Line up the def - end pairs
Line up the if - end pairs
Indent the code “within” the if-
end and def-end pairs.


```
1 class OneController < ApplicationController.  
2  
3   def index.  
4  
5  
6   def about.  
7   end.  
8  
9   def contact.  
10  end.  
11  
12  def thanks.  
13    logger.info "Welcome to the thanks action in the controller".  
14    logger.info params[:yourname].  
15    if params[:yourname] && params[:yourname] == "Chuck".  
16      @barcelona = "Membership approved.".  
17    else.  
18      @barcelona = "Membership pending".  
19    end.  
20  end.  
21  
22 end.
```


Lining things up makes it easier to see your mistakes - if you are stuck struggling with a “kEND” error - don’t just try to fix it by throwing in more “end” statements - first neaten up a bit. It will take a minute or two and you will be much happier. As you neaten up - you might just come across the error.

```
1 class OneController < ApplicationController.  
2  
3   def index.  
4  
5  
6   def about.  
7   end.  
8  
9   def contact.  
10  end.  
11  
12  def thanks.  
13    logger.info "Welcome to the thanks action in the controller".  
14    logger.info params[:yourname].  
15    if params[:yourname] && params[:yourname] == "Chuck".  
16      @barcelona = "Membership approved.".  
17    else.  
18      @barcelona = "Membership pending".  
19    end.  
20  end.  
21  
22 end.
```

Stare at this for a while and see if you can spot the error....
See if you can find the “missing” end.
Count how many if/def/class lines there are and count how many end lines there are. The number should be the same.

```
1 class OneController < ApplicationController.  
2  
3   def index.  
4  
5  
6   def about.  
7   end.  
8  
9   def contact.  
10  end.  
11  
12  def thanks.  
13    logger.info "Welcome to the thanks action in the controller".  
14    logger.info params[:yourname].  
15    if params[:yourname] && params[:yourname] == "Chuck".  
16      @barcelona = "Membership approved.".  
17    else.  
18      @barcelona = "Membership pending".  
19    end.  
20  end.  
21  
22 end.
```

one_controller.rb:22: syntax error, unexpected \$end, expecting kEND



This is maddening - the stupid thing is looking for an end - and there is already an end there - what more does it want?

Wrong approach - “If you want an *%\$&#! “end” I will give you an end”

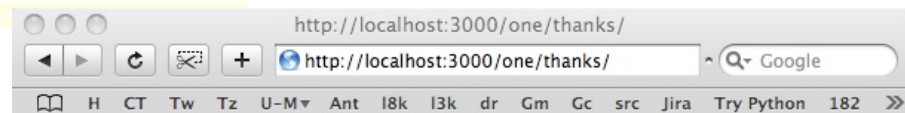
```

1 class OneController < ApplicationController.
2
3   def index.
4
5
6   def about.
7   end.
8
9   def contact.
10  end.
11
12  def thanks.
13    logger.info "Welcome to the thanks action in the controller".
14    logger.info params[:yourname].
15    if params[:yourname] && params[:yourname] == "Chuck".
16      @barcelona = "Membership approved.".
17    else.
18      @barcelona = "Membership pending".
19    end.
20  end.
21
22 end end.

```

We add an “end” at the end after that last “end”.
It wants an “end” - lets give it an “end”.

We hit refresh and the error goes away.
Success ???



One#thanks

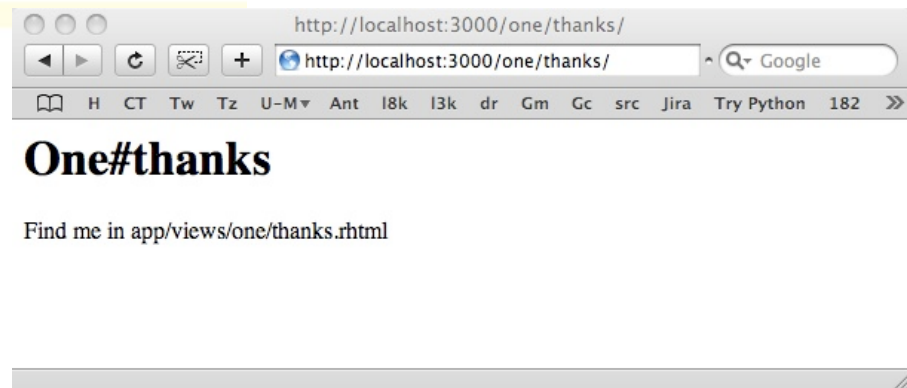
Find me in app/views/one/thanks.rhtml

```

1 class OneController < ApplicationController.
2
3   def index.
4
5
6   def about.
7   end.
8
9   def contact.
10  end.
11
12  def thanks.
13    logger.info "Welcome to the thanks action in the controller".
14    logger.info params[:yourname].
15    if params[:yourname] && params[:yourname] == "Chuck".
16      @barcelona = "Membership approved.".
17    else.
18      @barcelona = "Membership pending".
19    end.
20  end.
21
22 end end.

```

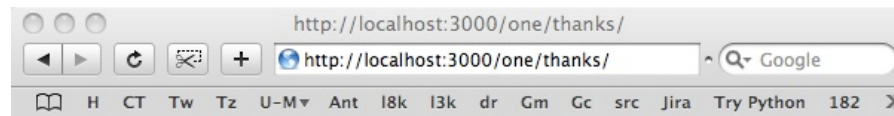
Not success at all - we fixed a syntax error but ruined the logic of our code. The if-end and def-end pairs tell Ruby/Rails what goes where - it defines blocks of code.



```

1 class OneController < ApplicationController.
2
3   def index.
4
5
6   def about.
7   end.
8
9   def contact.
10  end.
11
12  def thanks.
13    logger.info "Welcome to the thanks e
14    logger.info params[:yourname].
15    if params[:yourname] && params[:your
16      @barcelona = "Membership approved
17    else.
18      @barcelona = "Membership pending".
19    end.
20  end.
21
22 end end.

```



One#thanks

Find me in app/views/one/thanks.rhtml

If you solve this syntax error by adding an end at the end - your syntax error will go away- but the program will not function. Several hours later you will figure out that your “thanks” action is not actually executing because your thanks action is now part of the index action - because of where you put the end.

◇ one_controller.rb (/Users/csev/dev/rails_apps/app0/app/controllers/)

```
1 class OneController < ApplicationController.
```

```
2  
3 def index
```

```
4  
5  
6 def about.
```

```
7 end.
```

```
8  
9 def contact.
```

```
10 end.
```

```
11  
12 def thanks.
```

```
13   logger.info "Welcome to the thanks action in the controller".
```

```
14   logger.info params[:yourname].
```

```
15   if params[:yourname] && params[:yourname] == "Chuck".
```

```
16     @barcelona = "Membership approved. ".
```

```
17   else.
```

```
18     @barcelona = "Membership pending".
```

```
19   end.
```

```
20 end.
```

```
21  
22 end.
```

one_controller.rb:22: syntax error, unexpected \$end, expecting kEND

found end-of-file; expected 'rescue', 'ensure', 'end', 'else',

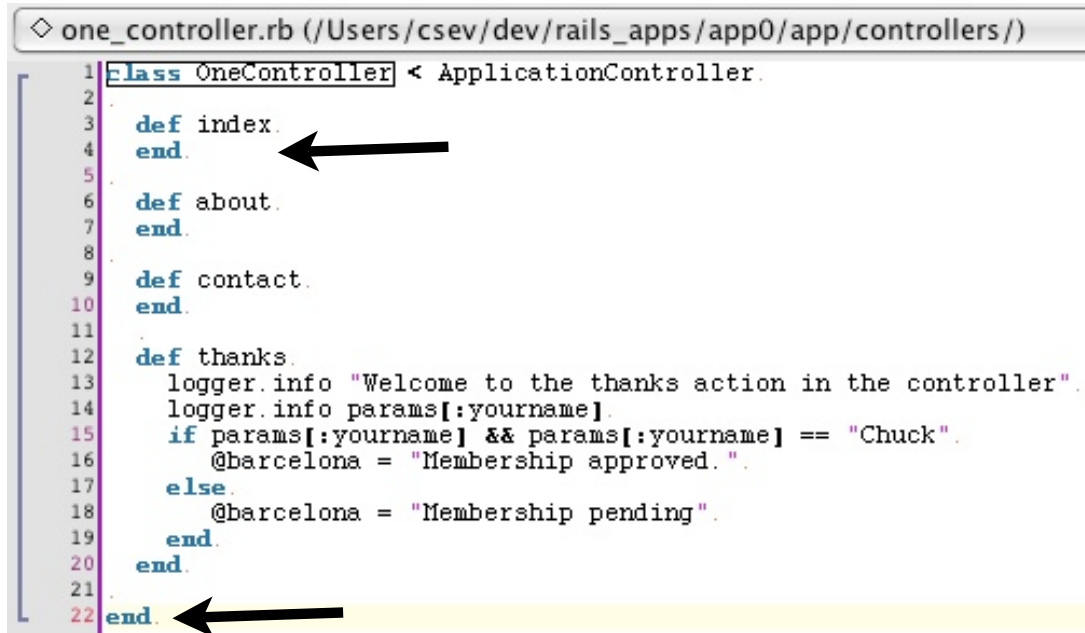
This is where the RubyPlugin in jEdit earns every penny of its free price. Put your cursor on the offending end. jEdit will count the if/def/class and end statements and show you which statement “matches” the offending end.

```
◇ one_controller.rb (/Users/csev/dev/rails_apps/app0/app/controllers/)
1 class OneController < ApplicationController.
2
3   def index
4
5
6   def about.
7   end.
8
9   def contact.
10  end.
11
12  def thanks.
13    logger.info "Welcome to the thanks action in the controller".
14    logger.info params[:yourname].
15    if params[:yourname] && params[:yourname] == "Chuck".
16      @barcelona = "Membership approved.".
17    else.
18      @barcelona = "Membership pending".
19    end.
20  end.
21
22 end.
```

found end-of-file; expected 'rescue', 'ensure', 'end', 'else',

We quickly see that the “end” on line 22 matched with the “def index” way back on line 3. And then we notice that there is a missing end on line 4 for the “def index”. The error was made on line 4 and noticed on line 22 because that was when the end-of-file was encountered and Ruby realised it was “missing” an end.


```
◇ one_controller.rb (/Users/csev/dev/rails_apps/app0/app/controllers/)
1 class OneController < ApplicationController.
2
3   def index.
4   end.
5
6   def about.
7   end.
8
9   def contact.
10  end.
11
12  def thanks.
13    logger.info "Welcome to the thanks action in the controller".
14    logger.info params[:yourname]
15    if params[:yourname] && params[:yourname] == "Chuck".
16      @barcelona = "Membership approved.".
17    else.
18      @barcelona = "Membership pending".
19    end.
20  end.
21
22 end.
```

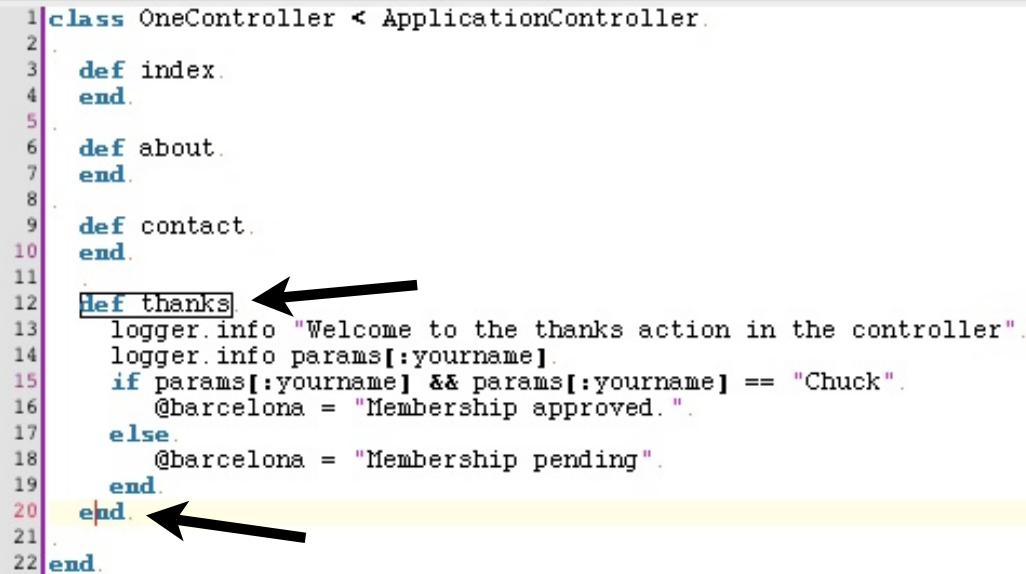


We put the “end” on line 4 in to close the “def index”.

Then we click on the “end” on line 22 and see that it properly matches the “class” statement on line 1.

◇ one_controller.rb (/Users/csev/dev/rails_apps/app0/app/controllers/)

```
1 class OneController < ApplicationController.  
2  
3   def index.  
4   end.  
5  
6   def about.  
7   end.  
8  
9   def contact.  
10  end.  
11  
12  def thanks  
13    logger.info "Welcome to the thanks action in the controller".  
14    logger.info params[:yourname].  
15    if params[:yourname] && params[:yourname] == "Chuck".  
16      @barcelona = "Membership approved.".  
17    else.  
18      @barcelona = "Membership pending".  
19    end.  
20  end.  
21  
22 end.
```



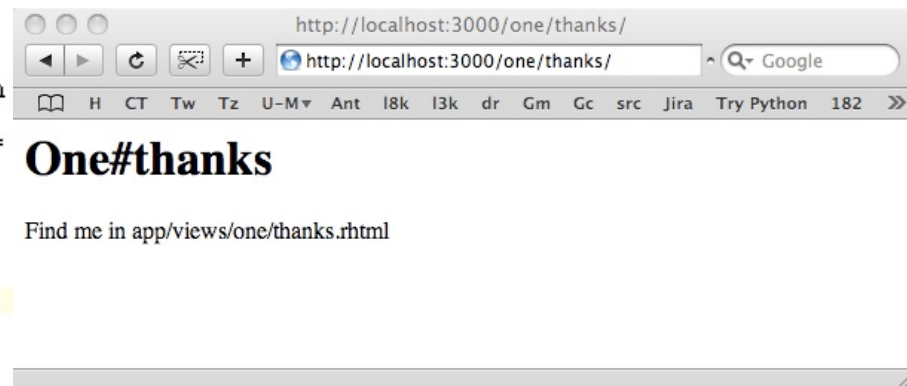
Once your program is syntax-error free - you can put the cursor on a def or class or end line and the Ruby Plugin will bracket the code between the start and end. This helps you keep track of your program logic.

```

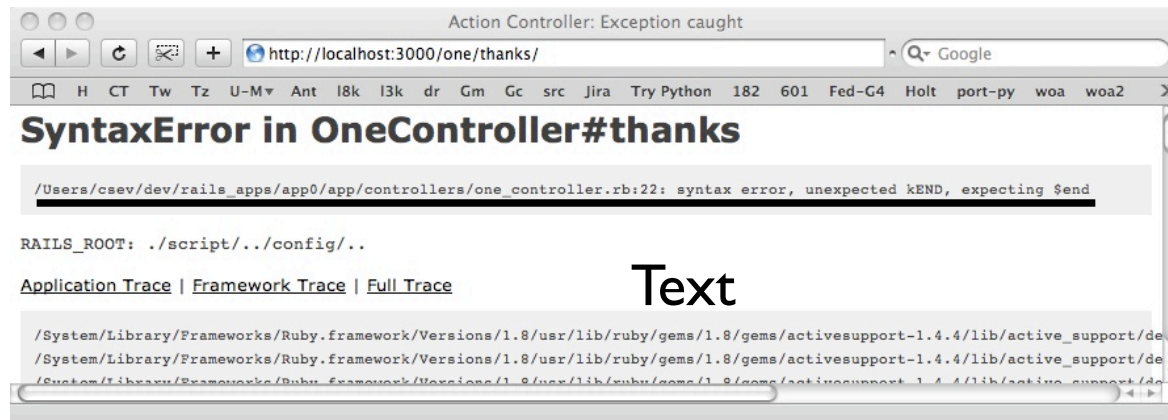
1 class OneController < ApplicationController.
2
3   def index.
4
5
6   def about.
7   end.
8
9   def contact.
10  end.
11
12  def thanks.
13    logger.info "Welcome to the thanks action in
14    logger.info params[:yourname].
15    if params[:yourname] && params[:yourname] ==
16      @barcelona = "Membership approved."
17    else.
18      @barcelona = "Membership pending".
19    end.
20  end.
21
22 end end.

```

one_controller.rb:22: syntax error, unexpected \$end, expecting kEND



If you solve this syntax error by adding an “end” at the end - your syntax error will go away - but the program will not function. Several hours or days later you will figure out that your “thanks” code has stopped executing because your thanks code has become part of the index action - because of where you put the “end”.



Too many end statements gives the same error as too few end statements.

If you have too many end statements - clicking on the end on line 22 will not show a bracket.

```
◇ one_controller.rb (/Users/csev/dev/rails_apps/app0/app/controllers/)  
1 class OneController < ApplicationController  
2  
3   def index  
4     end end  
5  
6   def about  
7     end  
8  
9   def contact  
10    end  
11  
12   def thanks  
13     logger.info "Welcome to the thanks action in the controller".  
14     logger.info params[:yourname].  
15     if params[:yourname] && params[:yourname] == "Chuck".  
16       @barcelona = "Membership approved.".  
17     else  
18       @barcelona = "Membership pending".  
19     end  
20   end  
21  
22 end
```

Traceback Information



The “trace back” information is only useful if it is in your code. All the other stuff is telling you about the inner workings of Rails - not much use to you - ignore it.

```
◇ one_controller.rb (/Users/csev/dev/rails_apps/app0/app/controllers/)  
1 class OneController < ApplicationController  
2  
3   def index  
4     end end  
5  
6   def about  
7     end  
8  
9   def contact  
10    end  
11  
12  def thanks  
13    logger.info "Welcome to the thanks action in the controller".  
14    logger.info params[:yourname].  
15    if params[:yourname] && params[:yourname] == "Chuck".  
16      @barcelona = "Membership approved.".  
17    else  
18      @barcelona = "Membership pending".  
19    end  
20  end  
21  
22 end
```

Dealing with kEND errors

```
one_controller.rb:22: syntax error, unexpected $end, expecting kEND
```

- Do not just start adding end statements until the error goes away - your code not have syntax errors - but it will probably also not work
- There **is** a reason and it is perfectly logical and with patience you will find it
- Let the RubyPlugin help by showing you the location of the error (red underline) and show you the beginning line that corresponds to the end line.

Summary

- Getting syntax errors can be frustrating for the beginning programmer
- Keeping your code neat and clean well indented helps avoid errors and when you make a mistake - it is easier to see the error.
- The error messages in the browser and in the logs can be pretty verbose. Ignore lines in trace backs that do not refer to parts of your program.
- In time you will realize that there are 4-5 typical typing mistakes you make and quickly recognize the error messages when you see them